

Oak Ridge National Laboratory

Storage at NCCS

Presented by
David Dillow

December 8, 2009



With whom am I speaking?

- How many in the audience are developing the scientific applications?
- How many in the audience are users of the scientific applications?

Overview

- WAN transfers
- HPSS
- Spider
- Quotas
- Best Practices

Moving your data over the WAN

- We have dedicated machines for this
 - dtn01.ccs.ornl.gov
 - dtn02.ccs.ornl.gov
- Spider center-wide file system is available there
- Transfers currently tuned for NERSC and ANL
 - Expect 100 to 200 MB/s
 - Other sites with large pipes should see similar performance
- More information on OLCF website
 - <http://www.nccs.gov/user-support/general-support/data-transfer>

BBCP

- Multiple streams to a single destination machine
- Available on all OLCF user systems
- Can be installed in your home directory if needed at remote
- May require firewall rule exceptions
- **Not** encrypted
- More information on OLCF website

<http://www.nccs.gov/user-support/general-support/data-transfer/bbcp>

GridFTP

- Preferred over BBP
- Multiple streams to multiple destination machines
- Optional encryption
- Can be automated with certificates
- Available on certain OLCF systems
- Can be installed in user directory, but more difficult
- More information on OLCF website

<http://www.nccs.gov/user-support/general-support/data-transfer/gridftp>

Protecting your data (HPSS)

- High Performance Storage System
- Accessible from all login nodes
- Accessible from data transfer nodes
- Over 8.2 PB storage in 16.5 M files stored
- Scratch storage space is just that
- If you care about your data, put it in HPSS
- Once copy of data is default, more can be requested

Using HPSS

- HSI
 - easy to use (FTP-like interface)
 - fine-grained control of parameters
 - works well for small numbers of large files
- HTAR
 - works like tar command
 - treats all files in the transfer as one file in HPSS
 - preferred way to handle large number of small files
- More information on OLCF website

<http://www.nccs.gov/computing-resources/hpss/use>

HPSS Best Practices

- If you care about your data, put it in HPSS
- When possible, consider transferring your files from the compute job (needs keytab access)
- Try to combine small files into larger ones
- Respect your fellow users – don't spawn large number of transfer processes on the login nodes
- Avoid needing to change the COS (expensive!) by letting the tools read from disk rather than streaming the output of a command

Spider, the OLCF center-wide file system



- “Spider” provides a shared, parallel file system for all systems
 - Based on Lustre file system
- Demonstrated bandwidth of over 240 GB/s
 - 120 GB/s currently provisioned to users
- Over 10 PB of RAID-6 Capacity
 - 13,440 1 TB SATA Drives
- 192 Storage servers
 - 3 Terabytes of memory
- Available from all systems via our high-performance scalable I/O network
 - Over 3,000 InfiniBand ports
 - Over 3 miles of cables
 - Scales as storage grows

Benefits of Spider

- Accessible from all major OLCF systems
 - Avoids data “islands”
 - No need to transfer data from simulation system to analysis system
 - Breaks the speed barrier of LAN transfers
 - Can use dedicated nodes to do WAN transfers
 - More friendly to your fellow users
 - Less likely to be caught by a system interrupt
- Accessible during maintenance windows
 - Spider remains accessible when Jaguar and JaguarPF are down
 - You can still get to your data!

Benefits of Spider (Continued)

- Unswept project spaces
 - 1 TB of space available for each project
 - Not backed up – use HPSS
- Higher performance HPSS transfers
 - XT Login nodes no longer the bottleneck

Drawbacks of Spider

- It's not all wine and roses – more like an expedition
- No one has run at this scale before
 - Spider is the largest, fastest Lustre file system in the world
- Activity on other systems can interact with your IO
- Tuning your use is important
- Pathological file system use can make for a bad day
 - We're working to find and help correct misbehaving applications
- We're all in this together – your help is appreciated

Current and Future File Systems at OLCF

System/Path	Size	Throughput	OSTs
Jaguar XT5			
(Future) /lustre/widow0	4198 TB	> 100 GB/s	672
/lustre/widow1	4198 TB	> 100 GB/s	672
Jaguar XT4			
(Future) /lustre/widow0	4198 TB	> 50 GB/s	672
/lustre/widow1	4198 TB	> 50 GB/s	672
/lustre/scr144	284 TB	> 40 GB/s	144
/lustre/scr72a	142 TB	> 20 GB/s	72
/lustre/scr72b	142 TB	> 20 GB/s	72
Lens			
(Future) /lustre/widow0	4198 TB	> 20 GB/s	672
/lustre/widow1	4198 TB	> 20 GB/s	672
Smoky			
(Future) /lustre/widow0	4198 TB	> 5 GB/s	672
/lustre/widow1	4198 TB	> 5 GB/s	672

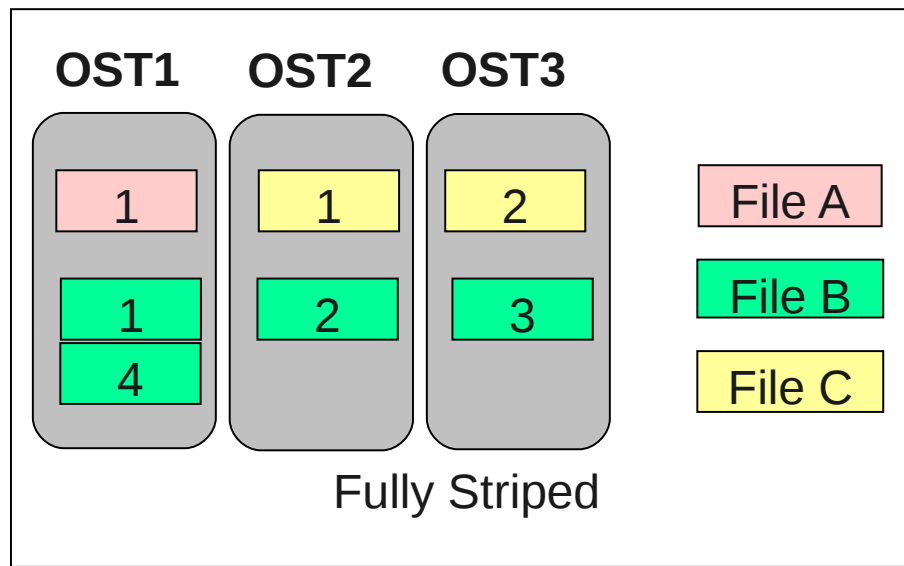
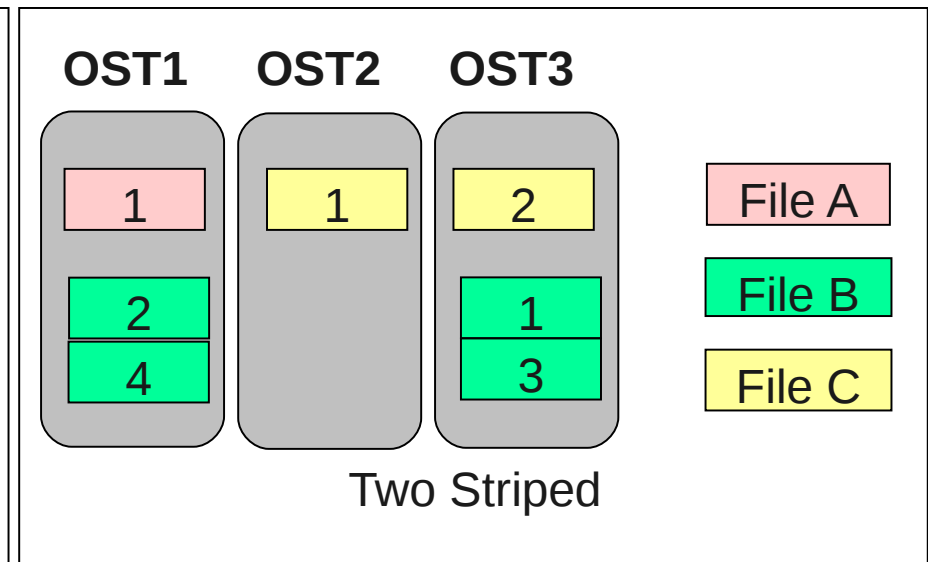
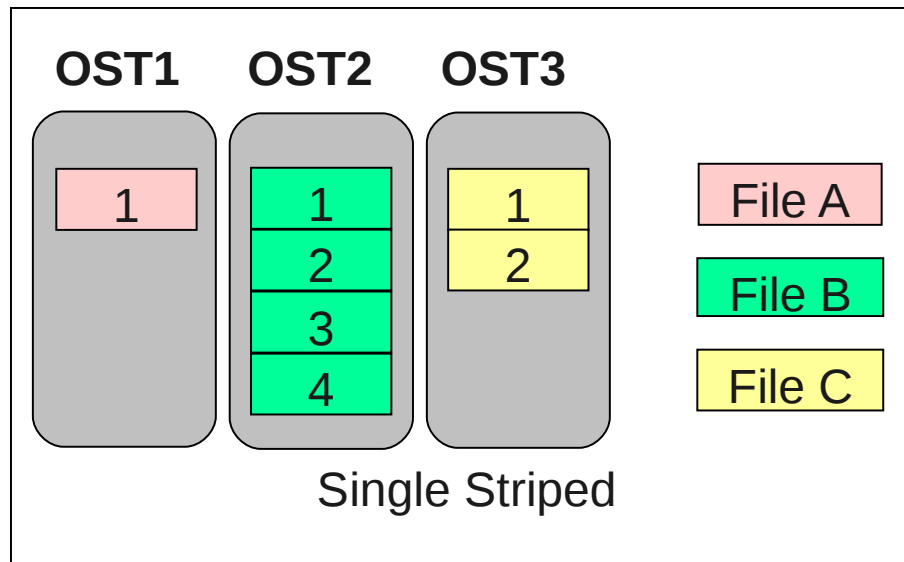
Quota policy

Area	Path	Quota	Swept?	Backups ?	Purge Policy
Home Directory	/ccs/home/\$USER	5 GB	No	Yes	1 month post-user
NFS Project	/ccs/proj/\$PROJ	50 GB	No	Yes	1 month post-project
Lustre Project	/tmp/proj/\$PROJ	1000 GB	No	No	1 month post-project
Spider Scratch	/tmp/work/\$USER	None	14 days	No	1 month post-user
Local Scratch	varies by system	None	14 days	No	1 month post-user
HPSS Home	/home/\$USER	2 TB 200 Files	-	-	3 months post-user
HPSS Project	/proj/\$PROJ	45 TB 4500 Files	-	-	3 months post-project

Lustre concepts

- Two types of servers
 - Metadata server (MDS)
 - Holds the directory tree
 - Stores metadata about each file (except for size)
 - Once file is opened, I/O to file does not involve the MDS
 - Object storage server (OSS)
 - Manages OSTs (think single disk/LUN)
 - OSTs hold stripes of the file contents
 - Think RAID0
 - Maintains the locking for the file contents it is responsible for

Lustre concepts (striping)



Lustre Best Practices (User)

- Use `lfs setstripe` in a safe manner
- Set striping appropriately for your use
- Choose stripe width for your application
- Avoid excessively large numbers of files in directories
- Avoid using `ls -l` repeatedly
- More information on website

<http://www.nccs.gov/user-support/general-support/file-systems/spider>

Lustre Best Practices (User)

- Use `lfs setstripe` in a safe manner
 - Always use the explicit options, not the positional ones
 - Avoid specifying a starting OST index
 - Use `-s` for stripe width (default is 1MB)
 - Can specify in bytes, kilobytes (k), megabytes (m), or gigabytes (g)
 - Use `-c` for stripe count (default is 4)
 - Not specifying an option keeps the current value

- Bad:

```
lfs setstripe $NAME 1m -1 16
```

- Good:

```
lfs setstripe $NAME -s 1m -c 16
```

Lustre Best Practices (User)

- Use `lfs getstripe` to check the striping on a file
- Example: extracting source code

```
# mkdir source
# lfs setstripe source -c 1
# cd source
# tar -x -f $STARFILE
```

- Example: fixing incorrect striping

```
# lfs setstripe newfile -c 16
# cp oldfile newfile
# rm oldfile
# mv newfile oldfile
```

Lustre Best Practices (User)

- Set striping appropriately for your use
 - Default stripe count is 4, but may not match your usage
 - Small files (< 250 MB) should use a single stripe
 - Large files accessed in parallel (single shared-file) should have a stripe count that is a factor of the number of writers (e.g. 20 vs 21 for 400 writers)
 - Cannot use more than 160 stripes currently

Lustre Best Practices (User)

- For single shared-file, choose per-writer data size as stripe width if possible
 - If each rank will write 256 MB, then use 256 MB as the stripe width
 - Minimizes lock contention
 - Liason can help determine best stripe size
 - May not always be possible to pick a winner

Lustre Best Practices (User)

- Avoid directories with excessive numbers of files in them
 - Excessive is a fuzzy number
 - > 1M definitely excessive
 - 100k probably excessive
 - 50k borderline
 - 10k sure, why not?

Lustre Best Practices (User)

- Avoid doing `ls -l` repeatedly
 - Especially in an excessively large directory!
 - If you are just looking to see if a file exists, use plain `ls`
 - Better yet – look for that file explicitly
 - Avoid options that sort by time stamp or add color to the listing

Lustre Best Practices (Developer)

- Open files read-only when possible
- Read small, shared files once
- Use a directory hierarchy to limit files in a single directory
- Use `access()`, not `stat()` to check for existence
- Avoid `flock()`
- Consider using libLUT or middleware I/O libraries
- Stripe-align I/O if possible

Lustre Best Practices (Developer)

- Open files read-only when possible
 - Fortran defaults to READWRITE if no ACTION is given
 - Fortran adds O_CREAT if opening file for writing
- O_CREAT requests an exclusive lock for the file (not contents)
 - Lock ping-pong championships when large job opens the file from all ranks at once

Lustre Best Practices (Developer)

- If all ranks need data from a single file, it is better to broadcast the contents than have everyone read it

Fortran example, sans error handling and assuming known file size:

```
CALL MPI_COMM_RANK(MPI_COMM_WORLD, my_rank, ierr)
IF (my_rank .eq. 0) THEN
    OPEN(UNIT=1, FILE=PathName, ACTION='READ')
    READ(1, *) buffer
ENDIF
CALL MPI_BCAST(buffer, SIZE, MPI_CHAR, 0,
               MPI_COMM_WORLD, ierr)
```

Lustre Best Practices (Developer)

- Use a directory hierarchy to limit files in a single directory
 - Opening a file currently keeps a lock on the parent directory for one message round-trip
 - Split directory up to avoid contention
 - For two level hierarchy, square root of the total number of files provides best balance

Lustre Best Practices (Developer)

- Use `access()`, not `stat()` to check for existence
 - Size is not kept on metadata server, so using `stat()` requires communication with each object storage server that has a portion of the file
 - `access()` only needs one request

Lustre Best Practices (Developer)

- Avoid flock()
 - $O(N^2)$ algorithm for number of lockers on file
 - Ok, if N is small
 - Does not scale to systems the size of Jaguar or JaguarPF

Lustre Best Practices (Developer)

- Consider using libLUT or middleware I/O libraries such as ADIOS
 - Extracting full performance from the file system requires knowledge of the environment
 - Maintaining performance during concurrent access from other users requires constant adaptation
 - Do you really want to write all of this?
 - And maintain it for multiple systems?

Lustre Best Practices (Developer)

- Stripe-align I/O if possible
 - Lustre is a POSIX-compliant file system
 - Overlapping writes are 'last-to-write wins'
 - This requires locking of the contents
 - Unaligned writes require obtaining locks from multiple servers

Questions?

- Contact info:

Galen Shipman

Group Leader, Technology Integration

865-576-2672

gshipman@ornl.gov